CS 1555

Lecture 26

**Access Paths and Indexing (continued)**

Index structure

- Performance accelerator for search on PK

- What about other files and attributes/fields?

- Single-level indexes

- Primary indexes

- Clustering indexes

- Secondary indexes

Point Access Methods (PAMs)

- A k-attribute record is envisioned as a point in k-dimensional space

- Can handle range queries

- Can handle both points and spatial objects

**Transaction Processing: Concurrency Control**

Two views of the system

- Application programmer’s view

- Sequence of SQL statements

- System developer’s view

- Sequence of reads and writes

Isolation

- The result of the execution of concurrent transactions is the same as if transactions were executed serially

- Serializability: operations may be interleaved, but execution must be equivalent to some sequential (serial) order of transactions

- Mechanism: concurrency control

Concurrency goal

- Execute a sequence of SQL statements so they “appear” to be running in isolation

- Simple solution: execute them in isolation

- But want to enable concurrency whenever it is safe

- High performance DBMS

- Benefit from modern architectures

Anomalies

- Why is concurrency control needed?

- To avoid the following anomalies

- Lost update problem

- Dirty read problem

- Unrepeatable read problem (phantom read)

Three bad dependencies

- Lost update: Write-write interaction

- Dirty data: Write-read interaction

- Unrepeatable read: Read-write interaction

Conflicting operations

- A conflict happens if we have two operations such that:

- They belong to two different transactions

- They both operate on the same data item

- One of them is a write

- Two operations conflict if it matters in which order they are performed

- Non conflicting operations are called compatible

- A compatibility table shows which operations are compatible

Schedules

- When transactions are executing concurrently, the order of execution of operations from all transactions is known as a schedule (or a history)

- A schedule S of n transactions is an ordering of the operations of the transactions

- For the purposes of concurrency control, we are mainly interested in the read (r) and write (w) operations, as well as commit (c) and abort (a) operations

Concurrency control schemes

- Lock-based CC schemes

- Two-phase locking (IBM DB2, SQLServer)

- Multiversion (Oracle, SQLServer)

- Timestamp-based

- Optimistic CC & certifiers

Lock based concurrency control

- Locking is the most common synchronization mechanism

- A lock is associated with each data item in the database

- A lock on item X indicates that a transaction is performing an operation (read or write) on X

- A transaction can issue the following operations on X

- read\_lock(x): X is read-locked by T

- Shared lock: other transactions are allowed to *read* x

- write\_lock(x): X is write-locked by T

- Exclusive lock: single transaction holds the lock on x

- unlock(x)

Basic two-phase locking (2PL)

- A scheduler following the 2PL protocol has two phases

- A growing phase

- Whenever the scheduler receives an operation on any item, it must acquire a lock on that item before executing the operation

- No locks can be released in this phase

- A shrinking phase

- Once a scheduler has released a lock for a transaction, it cannot request any additional locks on any data item for this transaction

Rigorous 2PL or industrial strict 2PL

- The growing phase: transaction request locks just before they operate on a data item

- The growing phase ends at commit time

Deadlocks

- A deadlock occurs when two or more transactions are blocked indefinitely

- This happens because each holds locks on data items on which the other transaction(s) attempt to place a conflicting lock

- Necessary conditions for deadlock situations: mutual exclusion, hold and wait, no preemption, circular wait

Deadlock handling schemes

- Deadlock avoidance

- Deadlock prevention

- Deadlock detection and resolution

Multiversion concurrency control

- The DBMS keeps a list of versions for each x

- Version xi means the version of x produced by a write on x by transaction Ti

- Each write(x) produces a new version of x

- When the scheduler receives a R(x), it must decide which version of x to read

- If a transaction T is aborted, any version it created is destroyed

- If a transaction T is committed, any version it created becomes available for other transactions

**Transaction Recovery**

Atomicity and durability

- Atomicity: transactions may abort (rollback)

- Durability: what if DBMS stops running?

Goal of recovery

- When a transaction T commits, make the updates permanent so they can survive subsequent failures

- When a transaction T aborts, obliterate any updates on data items and obliterate the effects of T on other transactions

- When the system crashes after a system/media failure, bring the database to its most recent state

Recovery actions

- Undo action: required for atomicity

- Undoes all updates on the stable storage by an uncommitted transaction

- Redo action: required for durability

Recovery techniques

- Undo/redo algorithm

- Undo/no-redo

- No-undo/redo (logging with deferred updates)

- No-undo/no-redo (shadowing)

Logging

- A log or journal is a sequence of records which represent all modifications to the database in the order in which they actually occurred

- Log records may describe either physical changes or logical database operations

- Physical log contains information about the actual values of data items written by transactions

- Logical log represents higher level operations

Cost of recovery

- To restart, we need to scan the entire log

- The restart operation will be prohibitively slow

- The log file may become very long and may not fit on disk

- Observation: most of the transactions that need to be redone have already written their updates

Garbage collection & checkpoints

- Recycling space in the log occupied by unnecessary info

- The amount of work restart has to do after a system failure can be reduced by checkpointing

- Force the updates that have been performed up to a certain time to materialize in the database

- Checkpoint record: include a list of transactions that were active at checkpoint time